Necessary conditions for designing secure stream ciphers with the minimal internal states

Vahid Amin Ghafari[1(✉)], Honggang Hu[1], and Mohammadsadegh Alizadeh[2]

[1]Key Laboratory of Electromagnetic Space Information, Chinese Academy of Sciences, University of Science and Technology of China, Hefei, China, 230027
[2]Department of Electrical Engineering,Sharif University of Technology
vahidaming@mail.ustc.edu.cn,hghu2005@ustc.edu.cn,alizadehms@ee.sharif.ir

Abstract. After the introduction of some stream ciphers with the minimal internal state, the design idea of these ciphers (i.e. the design of stream ciphers by using a secret key, not only in the initialization but also permanently in the keystream generation) has been developed. The idea lets to design lighter stream ciphers that they are suitable for devices with limited resources such as RFID, WSN.
We present necessary conditions for designing a secure stream cipher with the minimal internal state. Based on the conditions, we propose Fruit-128 stream cipher for 128-bit security against all types of attacks. Our implementations showed that the area size of Fruit-128 is about 25.2% smaller than that of Grain-128a. The discussions are presented that Fruit-128 is more resistant than Grain-128a to some attacks such as Related key chosen IV attack. Sprout, Fruit-v2 and Plantlet ciphers are vulnerable to time-memory-data trade-off (TMDTO) distinguishing attacks. For the first time, IV bits were permanently used to strengthen Fruit-128 against TMDTO attacks. We will show that if IV bits are not permanently available during the keystream production step, we can eliminate the IV mixing function from it. In this case, security level decreases to 69-bit against TMDTO distinguishing attacks (that based on the application might be tolerable). Dynamic initialization is another contribution of the paper (that it can strengthen initialization of all stream ciphers with low area cost).

Keywords: Stream Cipher, Ultra-lightweight, Lightweight, NFSR, LFSR, Hardware Implementation, Cryptographic Primitive

1 Introduction

There are so many devices that should be connected to each other safely. A big part of the devices has limited resources, and they need cheap equipment for privacy (such as RFID, WSN, and mobile communication). One of the best tools is stream ciphers with the minimal internal state for privacy. One old and famous rule tells that the internal state size should be at least twice of the security level in the stream ciphers. Sprout was proposed with a minimal internal state in FSE 2015 (the size of FSR in Sprout was 80 bits) [3]. The design idea of Sprout was

based on using the key not only in the initialization but also in the keystream generation. In other words, the key used as an internal state. It was an excellent idea because the keys should save for reused by others IV (in most of the real world stream ciphers). Thus, stream ciphers were designed with smaller area size.

A short while after the introduction of Sprout, many attacks were published against it [12, 14, 16, 5, 10, 20]. The key bits did not exploit correctly in the round key function. An important principle is that an internal state (here including key bits) should participate in internal state updating continually. The same impact ratio of all sections of the internal state on keystream production and internal state updating is critical. The key bits had little effect on the internal state updating in Sprout [2]. For compensating this problem in Plantlet cipher with 80-bit security, the size of FSR was increased to 101 bits. This change showed that the main challenge, i.e. design stream ciphers with internal state almost equal to the security level, has been forgotten. Furthermore, Plantlet suffers the insecure design similar Sprout (i.e. unbalanced participations of key as a part of the internal state in the keystream productions) [18]. A TMDTO distinguishing attack was proposed against Plantlet [11].

Fruit-v2 was another stream cipher with the minimal internal state. The size of FSR in Fruit-v2 was 80 bits, and it had 15 bits counter [2]. Two attacks were proposed to Fruit-v1 with 80-bit security, and it has been updated [7, 11, 2]. Fortunately, the main structure of Fruit-v2 is resistant against all attacks except TMDTO distinguishing attack [11]. The attack needs $2^{50.2}$ known keystream, $2^{50.2}$ memory and $2^{44.5}$ encryption for distinguishing the keystream of Fruit-v2 from truly random sequences. It is obvious that the attack is not practical (especially for low power hardware environment). Nevertheless, we propose necessary conditions that strengthen the design of stream ciphers with the minimal internal state against all types of attacks. In [11] a new idea was suggested to strengthen the ciphers against TMDTO distinguishing attack. The new idea was to use IV bits not only in the initialization but also in the keystream generation. In other words, the new idea was permanently using the IV (similar to the key) in the internal state updating and keystream production [11]. It is obvious that key should be saved for reuse by others IVs in the most real world stream ciphers. Thus, the using of the key does not impose any area in hardware (except for accessing the key bits).

Also permanently using IV does not impose any area in many applications. One famous example is $A5/1$ stream cipher in GSM system. Allowed IVs are 22-bit frame numbers that only one time are produced for every key (reuse the same IV is strictly prohibited with the same key in all stream ciphers). A 22-bit memory is dedicated to storing the IV (i.e. frame numbers) that it increases in every loading sequentially. It is a general structure that parameter(s) (e.g. packet number) is stored in memory as IV. Parameter(s) that is sequentially produced is new IVs. In this structure, there is no vulnerability regard to producing the same IV twice with the same key. In the cases that IVs are produced from some different parameters of the system, it is still possible that IV bits would

be available (from their sources) for permanently using in ciphers [40]. In these cases, the existence of a mechanism for prevention of production of the same IV under the same key is compulsory.

In situations that IV bits are not permanently available during the keystream production, we can suggest based on two important points that it is possible to ignore the IV mixing function in the keystream generation step (i.e. eliminate the round IV function from Fruit-128). First point, designers of Fruit-v2 and Plantlet ruled out TMDTO distinguishing attacks against ultra-lightweight ciphers (they stated that the attack needs so many resources while it is only distinguishing attacks). The second point, it is possible that the TMDTO distinguishing attacks would be impossible based on the application or environment of a cipher [11]. We will show that if we ignore the round IV function from Fruit-128, a TMDTO distinguishing attack is applicable with $2^{76.1}$ (known) bits of keystream, 2^{69} times encryption and 9,261,023,232 terabytes memory. Note that the attack only can distinguish the keystream of the cipher from truly random sequences.

Ultra-lightweight Stream cipher that Permanently Uses Key (USPUK): A stream cipher which uses key not only in the initialization but also permanently in the keystream generation and its GE (gate equivalents) is less than 80% of the hardware implementation of a Grain family member with the same security level.

Grain-v1 and Grain-128a are famous and concrete lightweight stream ciphers with 80 and 128-bit security, respectively [13, 1]. As these ciphers are secure and lightweight, these are a suitable basis for comparison. It is obvious that a hardware implementation should be in the same condition and GE of Grain-128a should be considered excluding authentication section. Sprout, Fruit-v2, and Plantlet are USPUK.

It is unacceptable in a USPUK that, the number of clocks in the initialization is much more than that of a Grain family member with the same security level. In Sprout and Plantlet cipher, the number of the initial clocks is 320 while it is only 160 clocks for Grain-v1. As Sprout and Plantlet were designed for devices with insufficient resources, it is not tolerable. Too many initial clocks were done to compensate the weak initialization in Sprout and Plantlet. The maximum number of clocks in Fruit-128 is 278 while it is 256 clocks for Grain-128a. We propose a dynamic initialization that is suitable for USPUKs. As the internal state updating function is bijective in stream ciphers, it is very simple to retrieve the key from the known internal state. Although internal state updating function is bijective in Fruit-128, retrieving the key bits is not as simple as the clock back. The dynamic initialization idea is also useful to all stream ciphers. It can increase security margin against key recovery attack from a known internal state with the small cost of the area in all stream ciphers.

USPUKs have been considered recently, and some papers were published about them. It seems that the necessary design conditions of these ciphers are clear now.

Necessary conditions for designing secure USPUK:

1- Internal state size should be twice of the security level in every situation (e.g. under fixed key).

2- All of the internal state bits (including key bits) should be independently affected on the internal state updating almost by the same ratio.

3- It is unacceptable to leave variable-key attacks (a resistant initialization should be used in USPUK). We will discuss in the next section about conditions.

The paper is organized as follows. The necessary design conditions of USPUKs are discussed. Then, Fruit-128 as an instance of secure USPUK will be presented. We discuss that Fruit-128 is resistant to known attacks. Finally, we show the results of the hardware implementation.

2 The necessary design conditions of USPUKs

There are four important spaces. The minimum size of these spaces specifies the security level margin. The half of the minimum size is the upper bound of security level against TMDTO attacks in every stream cipher. These are 1- the space of the key-IV 2- the space of the internal state after key-IV loading 3- the space of the internal state before the producing first keystream 4- the space of the internal state after the producing so many keystreams. In all possible conditions (e.g. for a fixed key), the upper bound of the security level against TMDTO attacks is the minimum size half of these spaces. If we consider a situation that key is fixed in Sprout, Fruit-v2 and Plantlet ciphers, all of them are vulnerable against TMDTO distinguishing attacks [11]. For example in Plantlet cipher, the space of the key and IV is 170 (i.e. 80+90) bits, but the space of the internal state is 108 (i.e. 40+61+7) bits. Therefore, the security level is 54-bit (i.e. 108/2) against distinguishing TMDTO distinguishing attacks in Plantlet. The claimed security (and the size of key-IV i.e. 170/2) stated that the security level should be 80-bit against TMDTO distinguishing attacks. Each 2^{62} (i.e. $2^{170}/2^{108}$) different key-IVs produces the same internal state and same keystream. An attacker supposes the key is fixed and he produces 2^{54} keystreams in Plantlet. He saves keystreams on a table. If he produces again 2^{54} keystreams with the same key and different IVs and search on the table, he can find at least one collision based on the birthday paradox (distinguishing attack) [11]. The attacker finds two different IVs that under the same key (but in a different clock) arrive at the same internal state. The attack on Plantlet was successfully applied because the internal state size is not twice of the security level with a fixed key. The above discussion showed that the first condition (i.e. internal state size should be twice of the security level under every fixed key) should be considered in the design of a USPUK.

The second condition is that all of the internal state bits (including key bits in a USPUK) should be independently affected on the internal state updating (and keystream generation) almost by the same ratio. It is obvious that if a part of internal state participates on the internal state updating much less than other parts, the stream cipher is vulnerable to many attacks (similar: divide-and-conquer and guess-and-determine attacks). This condition did not consider

in the design of Sprout, and many attacks were proposed it [12, 14, 16, 5, 10, 20]. In Plantlet cipher, this vulnerability was compensated by the increasing the size of FSR. The increasing has two problems. The first is increasing the area size and power consumption and the second one is oblivion of the initial challenge of the design (i.e. design stream ciphers with internal state almost equal to the security level). In Fruit-v2, this condition was carefully considered, and its main structure has been resisted against all types of attacks (except a trivial distinguishing attack that it was also applied to Planet [11]).

The last condition is about initialization procedure. Designers of Plantlet (and also Sprout) cipher mentioned that variable-key attacks (including related-key attacks) are out of the scope of Plantlet security (as we consider that the key is fixed, variable-key attacks (including related-key attacks) attacks are also out of scope[18]). This is while one of the main innovations of the paper is designing a cipher for efficiently using EEPROM for storing key bits. They suggested EEPROM because changing key is easy on EEPROM (they stated that EEPROM in a USPUK helps flexible key management) [18]. It is wonderful that the authors of a published paper in FSE 2017 forgot their main contribution and refused a weakness while it is directly related to their contribution. It seems that Plantlet is vulnerable to related-key attack such as Sprout (because the initialization and structure of Plantlet are similar to Sprout) [12, 19]. Thus variable-key attacks (and its special case i.e. related-key attacks) should be carefully considered. We think that every cipher should be secure against all types of attacks independent of its application.

3 Hardware aspect

Maybe it seems that access to some bits of the key at once (e.g. 6 bits of the key in Fruit-v2) is not efficiently possible. In the cases that one fixed key is sufficient forever (e.g. in RFID, WSN and SIM card), MROM and PROM are two suitable options for storing key bits. MROM and PROM allow to access to some bits of the key efficiently in the high speed at once. Although key management is limited in these cases, each user can burn his desirable key in a PROM. EEPROM is a type of non-volatile memory that allows flexible key management. If EEPROM is integrated into ASIC, there is not practically limitation on accessing to some bits of the key at once [18]. Therefore in the discussed cases, the number and how accessing the key bits does not significantly impact on the efficiency of a USPUK (although the influence on the area size is proportional to the number of access, it is completely practical in commercial cases).

The high number of accessing and non-sequential is very hard to commercial implementations in an external EEPROM with a serial interface. For the cases that we are interested in using external EEPROM, we should use a parallel interface for accessing memory. There are very different types of external EEPROM in the market. It seems that commercial implementation is possible in external EEPROM with parallel interface. External EEPROM suffers side channel at-

tacks. This vulnerability is as much as important that external EEPROM could be abandoned.

New design (i.e., Fruit-128) requires to access four key bits sequentially (while the round key function of Fruit-v2 uses six key bits randomly). In the round key function of Fruit-128, we divide 128-bit key to four independent 32-bit keys, and it needs that in every clock we access one bit of each 32-bit key cyclically. Thus it is possible to implement the round key function of Fruit-128 in all types of commercial memories.

4 The design of Fruit-128

The internal state consists of 65-bit LFSR $(l_t, ..., l_{(t+64)})$, 63-bit NFSR $(n_t, ..., n_{(t+62)})$, 10-bit counter $(Cr : (c_t^0, ..., c_t^9))$ and 7-bit counter $(Cc : (c_t^1 0, ..., c_t^{16}))$. A general view of Fruit-128 is presented in 1. Fruit-128 accepts 128-bit secret key (K: $(k_i, 0 \leq i \leq 127)$) and 128-bit public Initial Value, (IV: $(v_i, 0 \leq i \leq 127)$) as inputs. The maximum number of the keystream bits that can be produced from one key and IV is 2^{64} bits. IVs should be produced in a random way.

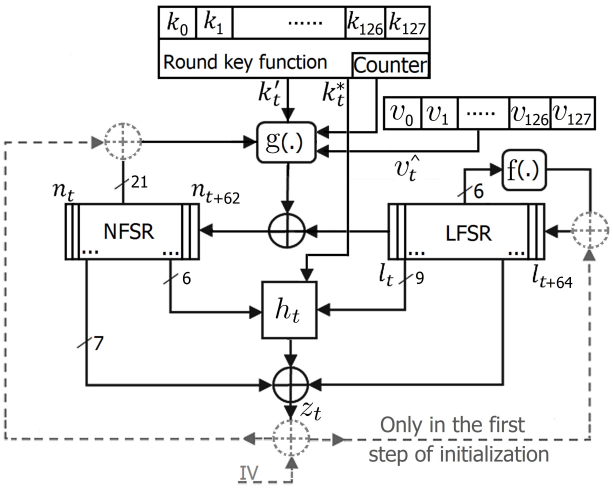

Fig. 1. The Block Diagram of Fruit-128

Now we explain each part of the cipher in details:

Counters. the first 10 bits of the counter (Cr) are allocated to the round key function, and the last 7 bits (Cc) are allocated to the initialization and keystream generation. These two counters work (count) cyclically and independently, i.e. the first counter $(c_t^0, ..., c_t^9)$ is increased one by one in each clock, and

also the second counter $(c_t^{10}, ..., c_t^{16})$ count from zero independently. These two counters increase at each clock, and work continually (i.e. after the first and second parts become all ones, counting from zeros to all ones again). Note that c_t^9 and c_t^{16} are LSB of the two counters, i.e. before the first clock, our counter is (0000000000000000). Then, after the first clock, our counter is (0000000010000001).

Round key function. we define some indexes of the key for using in the round key function. The indexes are 5 bits of Cr counter and they change in the each clock. We introduce $r = (c_t^0 c_t^1 c_t^2 c_t^3 c_t^4)$, $s = (c_t^5 c_t^6 c_t^7 c_t^8 c_t^9)$, $p = (c_t^2 c_t^3 c_t^4 c_t^5 c_t^6)$ and $q = (c_t^7 c_t^8 c_t^9 c_t^0 c_t^1)$. We combine 4 bits of the key to obtain the bits of the round key for **g** function (k'_t) and round key for $\boldsymbol{h}$ function(k_t^*) in each clock as follows.

$$\begin{aligned} k'_t =& k_r \cdot k_{(s+32)} \cdot k_{(p+64)} \oplus k_r \cdot k_{(p+64)} \oplus k_{(s+32)} \cdot k_{(q+96)} \\ & \oplus k_{(s+32)} \oplus k_{(p+64)} \oplus k_{(q+96)} \end{aligned} \tag{1}$$

$$\begin{aligned} k_t^* =& k_r \cdot k_{(s+32)} \cdot k_{(q+96)} \oplus k_r \cdot k_{(s+32)} \oplus k_{(p+64)} \cdot k_{(q+96)} \\ & \oplus k_r \oplus k_{(s+32)} \oplus k_{(p+64)} \end{aligned} \tag{2}$$

Round IV function. we define an index of the IV for using in the round IV function. The index is dependent on Cr counter and it changes in the each clock. We introduce $y = (c_t^3 c_t^4 c_t^5 c_t^6 c_t^7 c_t^8 c_t^9)$.

$$v_t^{\wedge} = v_y \tag{3}$$

g function. we use 1 bit of the counter, k'_t, $v_t^{\wedge}$ and 21 bits of the NFSR as variables of **g** function for clocking of the NFSR. The feedback function of the NFSR is as follows.

$$\begin{aligned} n_{(t+63)} = & k'_t \oplus l_t \oplus v_t^{\wedge} \oplus c_t^6 \oplus n_t \oplus n_{(t+10)} \oplus n_{(t+29)} \oplus n_{(t+50)} \\ & \oplus n_{(t+62)} \cdot n_{(t+60)} \oplus n_{(t+15)} \cdot n_{(t+40)} \oplus n_{(t+25)} \cdot n_{(t+46)} \\ & \oplus n_{(t+1)} \cdot n_{(t+3)} \oplus n_{(t+38)} \cdot n_{(t+35)} \oplus n_{(t+8)} \cdot n_{(t+19)} \cdot n_{(t+28)} \\ & \oplus n_{(t+48)} \cdot n_{(t+55)} \cdot n_{(t+57)} \cdot n_{(t+59)} \end{aligned} \tag{4}$$

f function. the feedback function of the LFSR is primitive. Hence it can produce a sequence with the maximum period. The feedback function of the LFSR is as follows.

$$l_{(t+65)} = l_t \oplus l_{(t+7)} \oplus l_{(t+33)} \oplus l_{(t+53)} \oplus l_{(t+61)} \oplus l_{(t+63)} \tag{5}$$

***h* function.** this function produces pre-output bits from the LFSR, key bits, and NFSR states as follows.

$$\begin{aligned} h_t = {} & k_t^* \cdot n_{(t+62)} \oplus k_t^* \cdot n_{(t+1)} \oplus l_{(t+1)} \cdot l_{(t+45)} \\ & \oplus l_{(t+9)} \cdot l_{(t+57)} \oplus n_{(t+37)} \cdot l_{(t+25)} \oplus n_{(t+11)} \cdot l_{(t+64)} \\ & \oplus l_{(t+29)} \cdot l_{(t+17)} \oplus n_{(t+61)} \cdot n_{(t+62)} \oplus n_{(t+2)} \cdot n_{(t+1)} \cdot l_{(t+49)} \end{aligned} \tag{6}$$

Output function. the output stream will be produced by 7 bits from the NFSR, 1 bit from the LFSR, and output of ***h*** function as follows.

$$\begin{aligned} z_t = {} & h_t \oplus n_{(t+3)} \oplus n_{(t+27)} \oplus n_{(t+47)} \oplus n_{(t+43)} \oplus n_{(t+19)} \\ & \oplus n_{(t+53)} \oplus n_{(t+60)} \oplus l_{(t+37)} \end{aligned} \tag{7}$$

Initialization of the cipher. we extend IV bits to the 150 bits by concatenating 22 bits to the end of it. 21 bit zeros and 1 bit one are concatenated to the end of IV, as follows.

$$IV' = v_0 v_1 v_2 ... v_{125} v_{126} v_{127} 000...001 \tag{8}$$

In the initialization procedure, key bits are loaded to the NFSR and LFSR from LSB to MSB (k_0 to n_0, k_1 to n_1, ..., k_{62} to n_{62}, k_{63} to l_0, k_{64} to l_1, ..., k_{127} to l_{64}). c_0^0,c_0^1, ..., c_0^{15},c_0^{16} are set to 0 in the first step of the initialization. The cipher is clocked 150 times, but before each clock, the XOR of the output bits and IV' bits is fed to the NFSR and LFSR (i.e. $z_i \oplus v'_i$, $0 \leq i \leq 149$ (as show in Figure. 1).

Then, in the second step of the initialization, we set all bits of Cr equal to LSB of the NFSR ($c_{150}^0 = n_{150}$,$c_{150}^1 = n_{151}$, ..., $c_{150}^8 = n_{158}$,$c_{150}^9 = n_{159}$), 6 bits of Cc equal to LSB of the LFSR ($c_{150}^{11} = l_{150}$,$c_{150}^{12} = l_{151}$, ..., $c_{150}^{15} = l_{154}$,$c_{150}^{16} = l_{155}$), and also l_{150} is set to 1 (for preventing that LFSR becomes all zeros before producing the first bit of the keystream).

Then, in the third step of the initialization, the cipher should be clocked until all bits of Cc become ones ($c^{10} = c^{11} = ... = c^{16} = 1$) without the feedback in the LFSR and NFSR (i.e. during the third step of the initialization the feedback of $z_i \oplus v'_i$ is disconnected from the LFSR and NFSR). The number of clocks in the third step of the initialization is a random number from 65 to 128 (i.e. the minimum and maximum number of clocks are 65 is 128 respectively).

5 The design criteria

Limitation for the producing keystream. the maximum produced keystream is 2^{65} bits in each initialization (because the period of the NFSR is a multiple of $2^{65} - 1$).

Round key function. there are two important criteria in this section. First, it should be able to provide the appropriate participation of all bits of the key in the internal states updating. Second, the round key function should be lightweight in hardware. The round key function produces 2×2^{10} different keys by involving 4 bits of the key. The 4 bits are changed in every clock uniformly. If an attacker can (by guessing the internal state and known keystreams) obtain some bits of k'_t and k^*_t, it is hard to obtain the key bits due to the unknown counter (unknown index of the key in the functions) and nonlinear round key function. k^*_t was multiplied by NFSR bits to make high degree monomials based on key bits in the keystream (for forward and backward clocking).

Round IV function. one bit of IV is selected in every clock by this function cyclically, and it is sent to **g** function. The counter specifies which IV bit is selected.

g function. the function that produces $n_{(t+63)}$ has been chosen in 21 variables of the NFSR with regard to the light implementation in the hardware (it is 29 variables in Grain-128a). If we suppose $k'_t \oplus v^{\wedge}_t \oplus c^6_t = 0$, the nonlinearity of **g** function will be $2^4 \times 64192$ and resiliency 3. The variables for the highest degree term have been chosen from n_t with $t > 47$ that, the degree of polynomials reaches the maximum possible degree in NFSR very soon.

f function. LFSR generates the sequences with the maximum period (the feedback polynomial is primitive). We are sure that the period of the LFSR and NFSR is at least $2^{65} - 1$ (because l_{150} is set to 1 after disconnecting the feedback of output from the LFSR). Some attacks were proposed to the Grain family and Sprout from this weakness (i.e. it is possible that the LFSR becomes all zeros after key-IV loading) [21, 5].

Dynamic initialization. it is essential to generate the monomials with the maximal degree in the initialization, and to distribute them in all bits of the LFSR and NFSR. We copy six bits of the LFSR to six bits of Cc counter (i.e. all bits of Cc are changed except c^{10}_{150}) in the second step of initialization, and we continue the clocking in the third step of the initialization until all bits of Cc become ones. As c^{10}_{150} becomes zero in the end of the first step of the initialization, the number of clocks is a random number from 215 to 278 in the whole of the initialization. We increase security margin against key recovery attacks from known internal state by this new idea with small cost of area. As the internal state updating function is bijective, it is very simple in most of stream ciphers to retrieve key from known internal state. Although internal state updating function is bijective in Fruit-128, but retrieving the key bits is not as simple as clock back.

Output function. The nonlinearity of $\boldsymbol{h}$ function is 16192. We add 8 linear terms to increase the nonlinearity to $2^{(8.16192)}$, and also to build a function with 7 resiliency. The best linear approximation of the output function has 8 terms with $2^{(-7.415)}$ bias. Regard to the minimal internal state of Fruit-128 and direct influence of the key bits on the keystreams, we used a more complicated function in output function than that of Grain-128a.

6 The resistance to known attacks

Only TMDTO distinguishing attack was successfully applied to all of the US-PUK (i.e. Sprout, Fruit-v2 and Plantlet [11]). Designers of Fruit-v2 and Plantlet ruled out TMDTO distinguishing attacks against their ciphers. They stated that the attack needs so many resources (e.g. $2^{50.2}$ (known) bits of keystream and 147 terabytes memory for the attack on Fruit-v2) and it can only distinguish the keystream of the ciphers from truly random sequences. Nevertheless, as the attack complexity is significantly smaller than that of brute force attack, we considered it in the design of Fruit-128. We discuss that Fruit-128 is resistant against all types of attacks (especially TMDTO attacks).

Time-Memory-Data Trade-off Attack. it is known that the size of the internal state should be at least twice of its security level in every stream cipher to resist against this attack. It means that the number of the possible internal states should be at least 2^{256} for Fruit-128 after initialization procedure (or after so many clocks) to resist against classical TMDTO attacks. The key, IV, LFSR, NFSR, and counter were used as the internal state in Fruit-128. Thus the size of the internal state is bigger than 256 bits in all conditions of Fruit-128. The size of the internal state of Grain-128a and Fruit-128 under a fixed key is 256 and 266 bits respectively (266 is obtained from the sizes of IV, FSR and Cr). Therefore the TMDTO distinguishing attack similar [11] is not applicable to Fruit-128. Note that for a fixed key and IV in Fruit-128 (because of the LFSR period), it is not possible that threre exist same internal states in two different clocks (i.e. the period of the internal state is at least 2^{65}).

Another condition for resisting a stream cipher to TMDTO attack is a good sampling resistance [6]. It means that an attacker cannot easily find the internal states for producing the keystreams with special pattern (e.g. the keystreams that start with 10 ones). An attacker should fix at least 7 variables of $\boldsymbol{h}$ function to linearize the output function and to easily identify the special internal states (it is at least 4 variables for Grain-128a). Hence it is suitable.

TMDTO Distinguishing Attack without Round IV Function. here we want to investigate that if the round IV function is eliminated from Fruit-128, how is possible to apply TMDTO distinguishing attacks to it. An attacker (under a fixed key and without IV mixing function in the keystream generation step) produces 2^{69} keystreams. Because the internal state of Fruit is 138 bits,

every keystream should be at least 138 bits. He saves keystreams in a searchable table. If he produces again 2^{69} keystreams with the same key and different IVs and search them on the table, he can find at least one collision based on the birthday paradox (distinguishing attack) [11]. The attacker finds two different IVs that under the same key (but in a different clock) arrive at the same internal state (and consequently the same keystream). Thus the data, time and memory complexity of the attack are $2^{69} \times 138$ bits of keystream, 2^{69} running of the cipher and $2^{69} \times 138$ bits memory respectively. Note that the attack only can distinguish the keystream of the cipher from truly random sequences.

Guess and Determine Attack. this attack is very important (because the FSR in Fruit-128 is short and a guess and determine attack was applied to Sprout [5]. If an attacker wants to produce only the 9 bits of the keystream, he needs to guess all bits of the FSR. We suppose that the attacker guesses some bits of FSR. He can clock one time forward and one time backward and obtains k_t^* and k_{t-1}^*. For next clocks, he needs the new bits that should be generated by $\mathbf{g}$ function. The output of $\mathbf{g}$ function is dependent on k_t'. If the attacker guesses the k_t' bits, the complexity of the attack becomes more than that of an exhaustive search attack after 9 clocks.

In another attack scenario, if an attacker considers the k_t' bits as unknown variables, after 11 clocks, the polynomial degree of keystream reaches 19 based on 24 key bit variables. The degree grows very fast and he cannot solve the equation system. He cannot identify wrong candidates of internal states (or obtain new values of internal states) before solving the equation system. Note that if the attacker supposes that $n_{(t+62)} \oplus n_{(t+1)}$ is zero for some clocks, k_t^* becomes ineffective on keystreams. In this situation, he should guess all bits of the FSR and Cr (i.e. 138 bits) to apply the attack. The attacker needs to clock at least 128 clocks (he obtains one equation for k_t' in every clocks). As the solving the nonlinear equation system is very hard, the complexity of the attack is more than that of an exhaustive search attack.

Linear Approximation Attack. if the NFSR and the output function are not chosen with suitable nonlinearity and suitable resiliency, the ciphers with Grain structure will be vulnerable to linear approximations attack [17]. We choose the NFSR and output function with suitable nonlinearity and resiliency. The best linear approximation of the output has $2^{(-7.415)}$ bias as follows (we suppose $k_t' \oplus c_t^6 \oplus v_t^\wedge = k_t^* = 0$).

$$z_t = n_{(t+3)} \oplus n_{(t+27)} \oplus n_{(t+47)} \oplus n_{(t+43)} \oplus n_{(t+19)} \oplus n_{(t+53)} \oplus n_{(t+60)} \oplus l_{(t+37)} \quad (9)$$

The best linear approximation of the NFSR feedback function has $2^{(-6.6)}$ bias as follows.

$$n_{(t+63)} = l_t \oplus n_t \oplus n_{(t+10)} \oplus n_{(t+29)} \oplus n_{(t+50)} \quad (10)$$

If we remove the NFSR variables between these two relations (by XORing the shifted linear approximation of the output), we obtain the following relation with $2^{(-72.3)}$ bias (by Piling-up Lemma).

$$\begin{aligned} &z_t \oplus z_{(t+10)} \oplus z_{(t+29)} \oplus z_{(t+50)} \oplus z_{(t+63)} = \\ &l_{(t+3)} \oplus l_{(t+27)} \oplus l_{(t+47)} \oplus l_{(t+43)} \oplus l_{(t+19)} \oplus l_{(t+53)} \\ &\oplus l_{(t+60)} \oplus l_{(t+37)} \oplus l_{(t+47)} \oplus l_{(t+66)} \oplus l_{(t+87)} \oplus l_{(t+100)} \end{aligned} \tag{11}$$

If an attacker tries to obtain a relation only based on the output bits by using the feedback function of LFSR, the bias of the relation will be too small (i.e. about $2^{(-72.3)}$). Thus, Fruit-128 is resistant to this attack.

Related-key Attack. designers of Plantlet cipher mentioned that related-key attacks are out of scope of Plantlet security (because the key is fixed), but when they suggested EEPROM for saving key, and they emphasized on EEPROM (because it is easy to rewrite a new key on EEPROM [18]), we think related-key attacks should be considered. There are vulnerabilities in the initialization of all members of the Grain family [15, 8] and Sprout [12, 19]. However, we propose a new scheme in the initialization procedure to strengthen Fruit-128 against this attack. We use the IV bits permanently in the keystream generation and also we did not load the IV bits in the internal state and did not combine the IV and key bits straightforward together. Therefore, Fruit-128 is resistant to this attack by exploiting new ideas and asymmetric padding in the IV bits loading.

Cube Attack. dynamic Cube attack was applied to Grain-128 [9, 4] because the degree of feedback of NFSR was small, i.e. 2. The degree of feedback of NFSR was increased to 4 in Grain-128a to response this weakness [1]. Because of the suitable initial clock number of Fruit-128 and the high degree of NFSR feedback, it is too hard to find any small degree multiplicative expression based on some bits of the IV in the Boolean function of the keystream. In Fruit-128, the length of LFSR and NFSR is shorter than that of Grain-128a, and also there are 23 variables in the output function of Fruit-128 while there are 17 variables in the output function of Grain-128a. Thus, it is acceptable that the degree of key and IV variables in the initialization of Fruit-128 grows faster than that of Grain-128a. As the minimum number of the clocks in the initialization of Fruit-128 (i.e. 215 clocks) is less than that of Grain-128a (i.e. 256 clocks), we implemented a Cube attack on Fruit-128. These results show that Fruit-128 (with 215 initial clocks) is resistant to Cube attack.

Algebraic Attack. this type of attack has not been applied to Grain family, but a special case of this attack was applied to Sprout [4]. Weak round key function made this weakness in Sprout. It is impossible for an attacker to apply the pure algebraic attack on Fruit-128 because the degree of polynomials in the internal state grows very fast. We discuss that a combination of a guess and determine attack with an algebraic attack is not applicable to Fruit.

If an attacker guesses bits of the NFSR, Cr and outputs of the round key function, then he obtains three equations in each clock (one from the keystream generation based on the LFSR bits and two equations from the round key function based on key bits). These equations are degree 2 and 3. It is not easy to solve the equation system, but we suppose that the attacker can solve equations of keystream and obtain 1 bits of the LFSR in each clock. In this scenario, he should guess at least 2×63 bits of the round key function (because it needs at least 63 clocks to use all bits of the key and there are 2 round key function outputs in every clock). Totally, the attacker should guess 63+10+64 bits and he should solve the nonlinear equation system for key). Thus, the computational complexity of the attack is more than that of an exhaustive search attack.

Weak key-IV. there are weak key-IVs in Grain family [21] and Sprout [5, 19]. It is possible that all bits of the LFSR become zeros immediately after initialization for some key-IVs. In this situation, the LFSR remains all zeros for all clocks, and NFSR statistical properties will become non-random. In this case, the period of the cipher is unknown, and the keystream is only dependent on NFSR bits, and the cipher is vulnerable. This situation is very important in Fruit-128 with regard to the short FSR. As we set l_{150} to 1 in the second step of the initialization (and the LFSR works independently after that), it is impossible that all bits of the LFSR become zeros in Fruit-128. Thus, there is no weak key-IV in Fruit-128.

7 Implementation Results

The design of ultra-lightweight ciphers is very important in the technology, while there are needs for lightweight ciphers in many fields such as RFID and WSN. Our goal was to design a secure USPUK for 128-bit security against all types of attacks. We tried to choose the functions and parameters based on minimum area size. We evaluated that the security margin was minimum to achieve ultra-lightweight cipher.

To get area size in hardware implementation for Fruit-128 and Grain-128a (without authentication section), TSMC 90 nm technology process is used and Synopsys Design Compiler Version C-2009.06-SP5 is used for synthesis and optimization.

We compare the area size of the hardware implementation of ciphers. The area size of Fruit-128 is significantly less than Grain-128a (without authentication section), as expected with regard to the length of the internal state (the internal state of Grain-128a is 256 bits for the FSR and 8 bits for the counter in the initialization, but for Fruit-128 is 128 bits for the FSR and 17 bits for the counter). Note that one GE is equivalent to the area of a 2-way NAND gate. 1 shows that the area size of Grain-128a is about 25.2% bigger than that of Fruit-128 in our results. It is obvious that a hardware implementation should be in the same condition and GE of Grain-128a should be considered excluding authentication section. We did not dedicate any GE to storing the key and IV bits. It was supposed that key and IV bits were saved for application purpose.

Table 1. The area size for Fruit-128 and Grain-128a (without authentication section) in hardware implementation

Cipher	Area size(GE)	Throughput (Kb/s)	Platform	Source
Grain-128a$^{\Omega}$ [1]	2146	100	-$^{\Omega}$	[1]
Grain-128a [1]	2001	100	90 nm CMOS	Our work
Fruit-128	1598	100	90 nm CMOS	Our work

$^{\Omega}$The estimated gate count in an actual implementation [1]

Key bits should be stored in the most applications for reuse with different IVs (implementations of Sprout, Fruit-v2, and Plantlet were done by this suppose on key bits [3, 2, 18]). As discussed before, usually IV bits are independently produced in encryption and decryption side sequentially from the parameter(s) of the system (e.g. from packet number in $A5/1$). Most of the system parameters should be saved in memories for updating and next using. Usually, the memories values are used as IV. New IVs and parameters are produced sequentially. Producing the same IV is impossible under the same key. In the cases that IVs are produced from some different parameters of the system, it is still possible that IV bits would be available (from their sources) for permanently using in ciphers. In these cases, the existence of a mechanism for prevention of production of the same IV under the same key is compulsory. As already mentioned it is possible to eliminate the round IV function from Fruit-128.

8 Conclusion

If we consider a situation that key is fixed in Fruit-v2 and Plantlet ciphers, all of them are vulnerable to TMDTO distinguishing attacks. In addition, although the number of clocks in the initialization of Plantlet is twice of the initial clocks of Grain-v1 (and this is not suitable for devices with limited resources), it seems that the initialization of Planet suffers variable-key attacks. In Plantlet proposal variable-key attacks was left because authors stated that key is fixed (this is not acceptable while one of the main design contributions was storing the key on EEPROM to rewrite new keys easily). Thus we proposed Fruit-128 that it is resistant against all types of attacks (especially to variable-key and TMDTO distinguishing attacks).

Fruit-128 has an obvious advantage in the applications that key and IV are permanently available in the ciphering system. The area size of Fruit-128 is about 25.2% smaller than that of Grain-128a in hardware implementation.

In situations that IV bits are not permanently available, it is possible to eliminate the IV mixing function from the keystream production step. We showed that if we ignore the round IV function from Fruit-128, a TMDTO attack is applicable for distinguishing the keystream from truly random sequences under a fixed key (with $2^{76.1}$ (known) bits of keystream, 2^{69} times encryption and 9,261,023,232 terabytes memory). The attack is unpractical and based on the

application might be tolerable (the security level decreases to 69 bits against distinguishing attack).

Round key function of Fruit-128 is lighter than that of Fruit-v2 and accessing to key bits on memory is sequentially (thus Fruit-128 can efficiently work on hardware). It is possible to redesign many of stream ciphers by design idea of Fruit-128 and achieve significantly smaller area size.

9 Acknowledgement

This work has been supported by CAS-TWAS President's Fellowship for International PhD program.

References

1. Ågren, M., Hell, M., Johansson, T., Meier, W.: Grain-128a: a new version of grain-128 with optional authentication. IJWMC 5(1), 48–59 (2011)
2. Aminghafari, V., Hu, H.: Fruit-v2: ultra-lightweight stream cipher with shorter internal state. IACR Cryptology ePrint Archive 2016, 355 (2016)
3. Armknecht, F., Mikhalev, V.: On lightweight stream ciphers with shorter internal states. In: FSE. Lecture Notes in Computer Science, vol. 9054, pp. 451–470. Springer (2015)
4. Aumasson, J., Dinur, I., Henzen, L., Meier, W., Shamir, A.: Efficient FPGA implementations of high-dimensional cube testers on the stream cipher grain-128. IACR Cryptology ePrint Archive 2009, 218 (2009)
5. Banik, S.: Some results on sprout. In: INDOCRYPT. Lecture Notes in Computer Science, vol. 9462, pp. 124–139. Springer (2015)
6. Biryukov, A., Shamir, A., Wagner, D.: Real time cryptanalysis of A5/1 on a PC. In: FSE. Lecture Notes in Computer Science, vol. 1978, pp. 1–18. Springer (2000)
7. Dey, S., Sarkar, S.: Cryptanalysis of full round fruit. IACR Cryptology ePrint Archive 2017, 87 (2017)
8. Ding, L., Guan, J.: Related key chosen IV attack on grain-128a stream cipher. IEEE Trans. Information Forensics and Security 8(5), 803–809 (2013)
9. Dinur, I., Shamir, A.: Breaking grain-128 with dynamic cube attacks. In: FSE. Lecture Notes in Computer Science, vol. 6733, pp. 167–187. Springer (2011)
10. Esgin, M.F., Kara, O.: Practical cryptanalysis of full sprout with TMD tradeoff attacks. In: SAC. Lecture Notes in Computer Science, vol. 9566, pp. 67–85. Springer (2015)
11. Hamann, M., Krause, M., Meier, W., Zhang, B.: Time-memory-data tradeoff attacks against small-state stream ciphers. IACR Cryptology ePrint Archive 2017, 384 (2017)
12. Hao, Y.: A related-key chosen-iv distinguishing attack on full sprout stream cipher. IACR Cryptology ePrint Archive 2015, 231 (2015)
13. Hell, M., Johansson, T., Meier, W.: Grain: a stream cipher for constrained environments. IJWMC 2(1), 86–93 (2007)
14. Lallemand, V., Naya-Plasencia, M.: Cryptanalysis of full sprout. In: CRYPTO (1). Lecture Notes in Computer Science, vol. 9215, pp. 663–682. Springer (2015)

15. Lee, Y., Jeong, K., Sung, J., Hong, S.: Related-key chosen IV attacks on grain-v1 and grain-128. In: ACISP. Lecture Notes in Computer Science, vol. 5107, pp. 321–335. Springer (2008)
16. Maitra, S., Sarkar, S., Baksi, A., Dey, P.: Key recovery from state information of sprout: Application to cryptanalysis and fault attack. IACR Cryptology ePrint Archive 2015, 236 (2015)
17. Maximov, A.: Cryptanalysis of the "grain" family of stream ciphers. In: ASIACCS. pp. 283–288. ACM (2006)
18. Mikhalev, V., Armknecht, F., Müller, C.: On ciphers that continuously access the non-volatile key. IACR Transactions on Symmetric Cryptology 2016(2), 52–79 (2017)
19. Roy, D., Mukhopadhyay, S.: Fault analysis and weak key-iv attack on sprout. IACR Cryptology ePrint Archive 2016, 207 (2016)
20. Zhang, B., Gong, X.: Another tradeoff attack on sprout-like stream ciphers. In: ASIACRYPT (2). Lecture Notes in Computer Science, vol. 9453, pp. 561–585. Springer (2015)
21. Zhang, H., Wang, X.: Cryptanalysis of stream cipher grain family. IACR Cryptology ePrint Archive 2009, 109 (2009)